

Méthodes de Partitionnement et d'apprentissage non supervisé

Classification Hiérarchique et Kmeans

Anne Badel, Frédéric Guyon & Jacques van Helden

2019-02-20

Partitionnement et apprentissage

- ▶ On a une **représentation** des données
 - ▶ sous forme de valeurs réelles=vecteur de
 - ▶ sous forme de catégories
- ▶ **Clustering**: on cherche a priori des groupes dans les données
- ▶ **Apprentissage**:
 - ▶ on connaît le partitionnement sur un jeu de données
 - ▶ on cherche le groupe (la classe) de nouvelles données

Partitionnement = Clustering

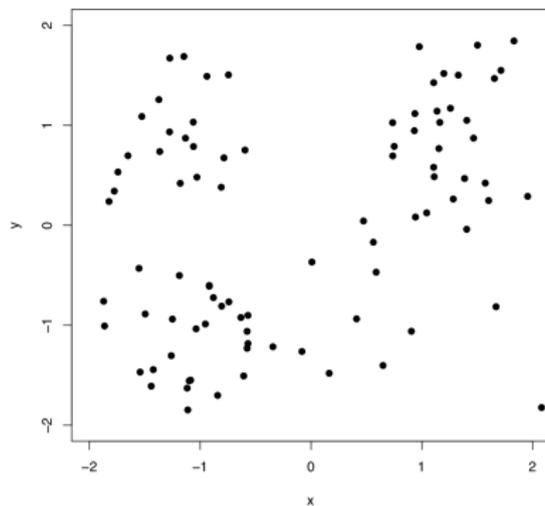


Figure 1: Y a-t-il des groupes ?

Partionnement = Clustering

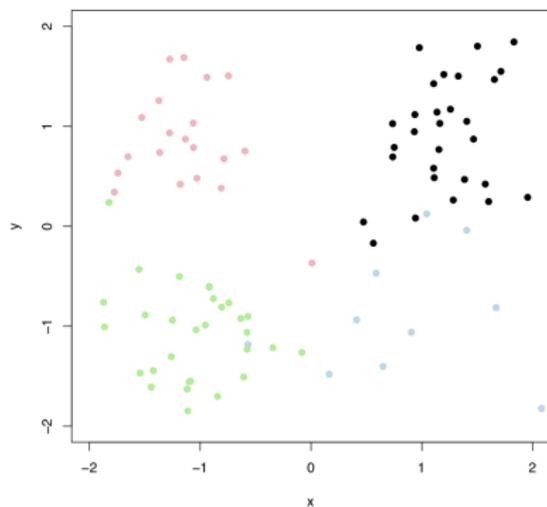


Figure 2: Oui, 4 groupes.

Apprentissage

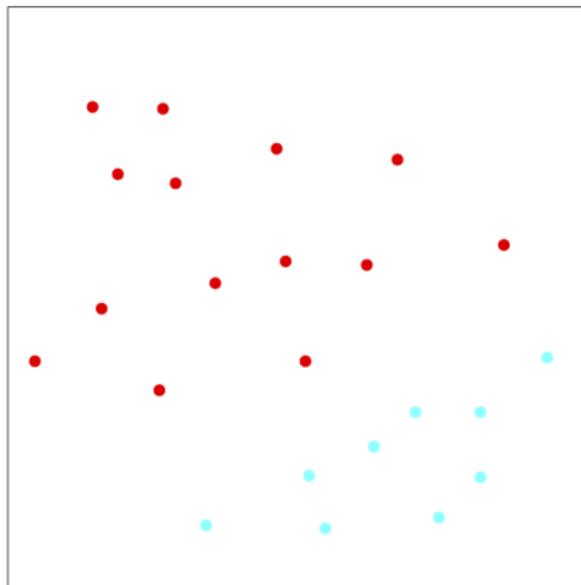


Figure 3: 2 groupes.

Apprentissage: Séparation linéaire

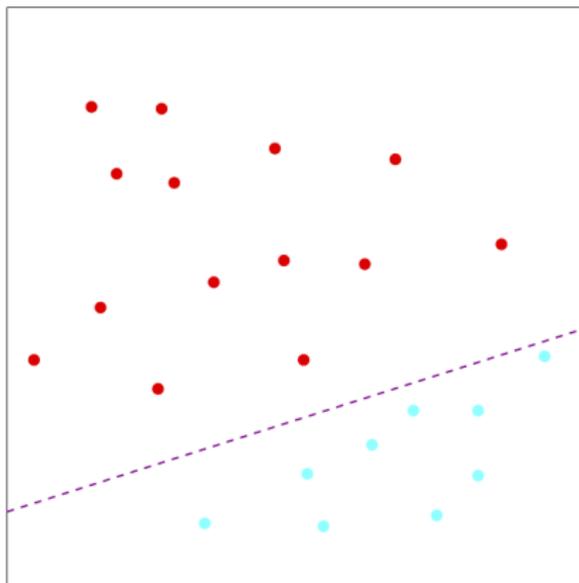


Figure 4: 2 groupes.

Méthodes

Trois grands principes de méthodes basées sur:

- ▶ La géométrie
- ▶ Les probabilités (statistique)
- ▶ Les graphes

En fait, trois façons de voir les mêmes algorithmes

Géométrie et distances

On considère les données comme des points de R^n (*)

- ▶ géométrie donnée par distances
- ▶ distances = dissimilarités imposées par le problème
- ▶ dissimilarités \rightarrow permettent visualisation de l'ensemble des points
- ▶ Détermination visuelle des groupes

(*) Espace Euclidien à n dimensions, où

- ▶ chaque dimension représente une des variables observées;
- ▶ un individu est décrit comme un vecteur à n valeurs, qui correspond à un point dans cet espace.

Les données

Ces données sont un classique des méthodes d'apprentissage

Dans un premier temps, regardons les données.

```
dim(mes.iris)
```

```
[1] 150  4
```

```
head(mes.iris)
```

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
1	5.1	3.5	1.4	0.2
2	4.9	3.0	1.4	0.2
3	4.7	3.2	1.3	0.2
4	4.6	3.1	1.5	0.2
5	5.0	3.6	1.4	0.2
6	5.4	3.9	1.7	0.4

Les variables

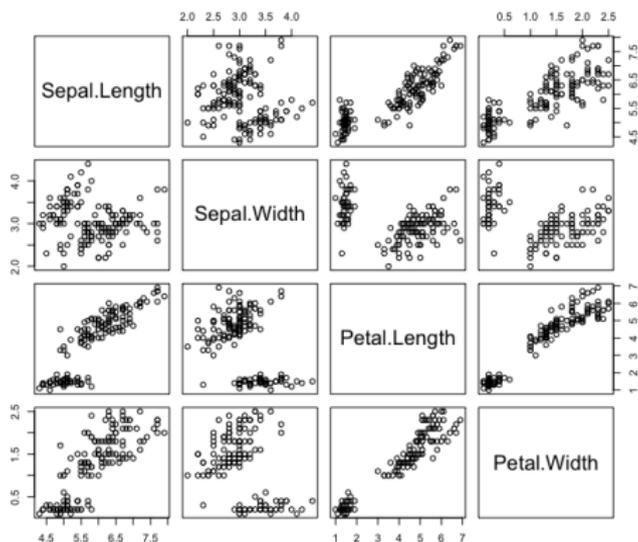
```
summary(mes.iris)
```

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
Min. :4.300	Min. :2.000	Min. :1.000	Min. :0.100
1st Qu.:5.100	1st Qu.:2.800	1st Qu.:1.600	1st Qu.:0.400
Median :5.800	Median :3.000	Median :4.350	Median :1.300
Mean :5.843	Mean :3.057	Mean :3.758	Mean :1.199
3rd Qu.:6.400	3rd Qu.:3.300	3rd Qu.:5.100	3rd Qu.:1.800
Max. :7.900	Max. :4.400	Max. :6.900	Max. :2.500

Visualisation des données

On peut ensuite essayer de visualiser les données

```
plot(mes.iris)
```



Cas d'étude : TCGA Breast Invasive Cancer (BIC)

- ▶ Présentation du cas d'étude (Jacques van Helden A COMPLETER)

TP : analyse de données d'expression

- ▶ TP clustering : [html][pdf] [Rmd]
- ▶ Première partie : chargement des données

Géométrie et distances

Sur la base d'une distance (souvent euclidienne)

- ▶ Partitionnement:
 - ▶ Moyennes mobiles ou K-means : séparation optimale des groupes connaissant le nombre de groupes
 - ▶ Méthode agglomérative ou hierarchical clustering
- ▶ Classification:
 - ▶ attribution K plus proches voisins (K Nearest Neighbor)
 - ▶ séparation linéaire ou non linéaire

Distances

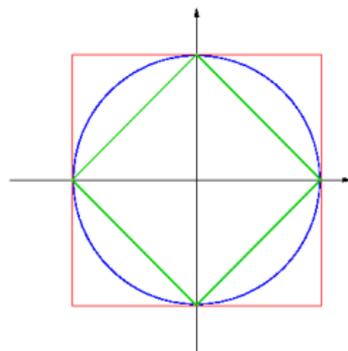
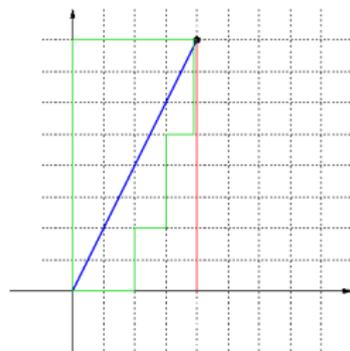
Définition d'une distance : fonction positive de deux variables

1. $d(x, y) \geq 0$
2. $d(x, y) = d(y, x)$
3. $d(x, y) = 0 \iff x = y$
4. **Inégalité triangulaire** : $d(x, z) \leq d(x, y) + d(y, z)$

Si 1,2,3 : dissimilarité

Distances utilisées dans R

- ▶ distance euclidienne ou distance L_2 : $d(x, y) = \sqrt{\sum_i (x_i - y_i)^2}$
- ▶ distance de manhattan ou distance L_1 : $d(x, y) = \sum_i |x_i - y_i|$
- ▶ distance du maximum ou L-infinis, L_∞ : $d(x, y) = \max_i |x_i - y_i|$



- distance euclidienne
- distance du maximum
- distance de Manhattan

Distances utilisées dans R

- ▶ distance de Minkowski l_p :

$$d(x, y) = \sqrt[p]{\sum_i (|x_i - y_i|^p)}$$

- ▶ distance de Canberra (x et y valeurs positives):

$$d(x, y) = \sum_i \frac{x_i - y_i}{x_i + y_i}$$

- ▶ distance binaire ou distance de Jaccard ou Tanimoto:
proportion de propriétés communes

Autres distances non géométriques (pour information)

Utilisées en bio-informatique:

- ▶ Distance de **Hamming**: nombre de remplacements de caractères (substitutions)
- ▶ Distance de **Levenshtein**: nombre de substitutions, insertions, deletions entre deux chaînes de caractères

$$d(\text{"BONJOUR"}, \text{"BONSOIR"}) = 2$$

- ▶ Distance d'**alignements**: distances de Levenshtein avec poids (par ex. matrices BLOSSUM)
- ▶ Distances d'**arbre** (Neighbor Joining)
- ▶ Distances **ultra-métriques** (phylogénie UPGMA)

Distances plus classiques en génomique

Comme vu lors de la séance 3, il existe d'autres mesures de distances :

- ▶ **Jaccard** (comparaison d'ensembles): $J_D = \frac{A \cap B}{A \cup B}$
- ▶ Distance du χ^2 (comparaison de tableau d'effectifs)

Ne sont pas des distances, mais indices de dissimilarité :

- ▶ **Bray-Curtis** (en écologie, comparaison d'abondance d'espèces)
- ▶ **Jensen-Shannon** (comparaison de distributions)

Remarque : lors du TP, sur les données d'expression RNA-seq, nous utiliserons le **coefficient de corrélation de Spearman** et la distance dérivée, $d_c = 1 - r$

Distances entre groupes

- ▶ **Single linkage** : éléments les plus proches des 2 groupes

$$D(C_1, C_2) = \min_{i \in C_1, j \in C_2} D(x_i, x_j)$$

- ▶ **Complete linkage** : éléments les plus éloignés des 2 groupes

$$D(C_1, C_2) = \max_{i \in C_1, j \in C_2} D(x_i, x_j)$$

- ▶ **Group average** : distance moyenne

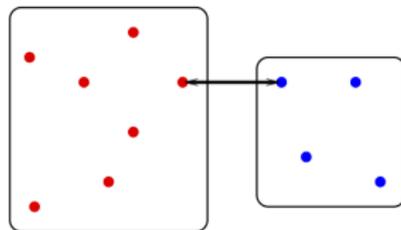
$$D(C_1, C_2) = \frac{1}{N_1 N_2} \sum_{i \in C_1, j \in C_2} D(x_i, x_j)$$

- ▶ **Ward**

$$d^2(C_i, C_j) = I_{intra}(C_i \cup C_j) - I_{intra}(C_i) - I_{intra}(C_j)$$

Distances entre groupes

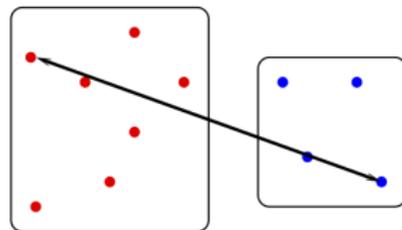
Single



Classe 1

Classe 2

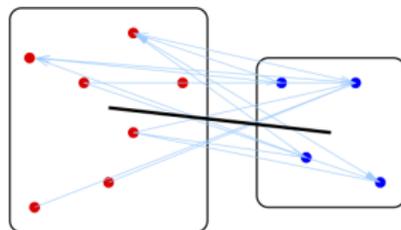
Complete



Classe 1

Classe 2

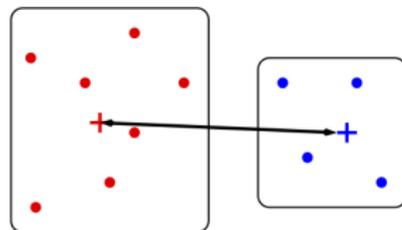
Average



Classe 1

Classe 2

Ward



Classe 1

Classe 2

Les données

Ces données sont un classique des méthodes d'apprentissage

Dans un premier temps, regardons les données

```
dim(mes.iris)
```

```
[1] 150  4
```

```
head(mes.iris)
```

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
1	5.1	3.5	1.4	0.2
2	4.9	3.0	1.4	0.2
3	4.7	3.2	1.3	0.2
4	4.6	3.1	1.5	0.2
5	5.0	3.6	1.4	0.2
6	5.4	3.9	1.7	0.4

```
str(mes.iris)
```

```
'data.frame':  150 obs. of  4 variables:  
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...  
 $ Sepal.Width  : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...  
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...  
 $ Petal.Width  : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.2 ...
```

```
summary(mes.iris)
```

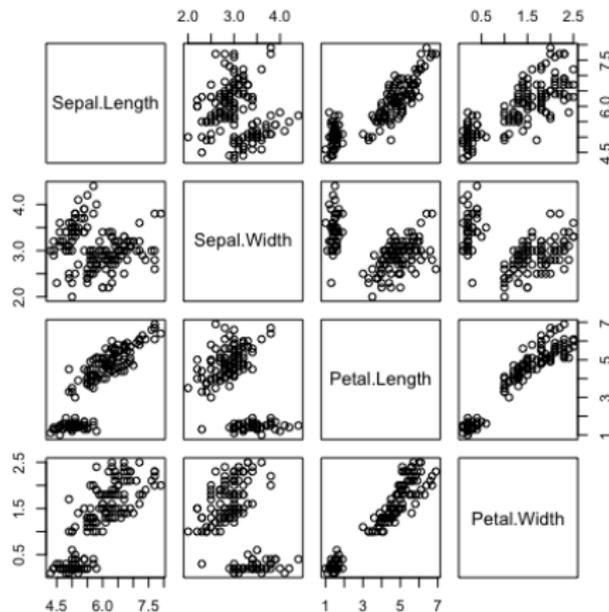
Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
Min. :4.300	Min. :2.000	Min. :1.000	Min. :0.100
1st Qu.:5.100	1st Qu.:2.800	1st Qu.:1.600	1st Qu.:0.100
Median :5.800	Median :3.000	Median :4.350	Median :1.050
Mean :5.843	Mean :3.057	Mean :3.758	Mean :1.053
3rd Qu.:6.400	3rd Qu.:3.300	3rd Qu.:5.100	3rd Qu.:1.300
Max. :7.900	Max. :4.400	Max. :6.900	Max. :2.500

Visualisation des données

On peut ensuite essayer de visualiser les données

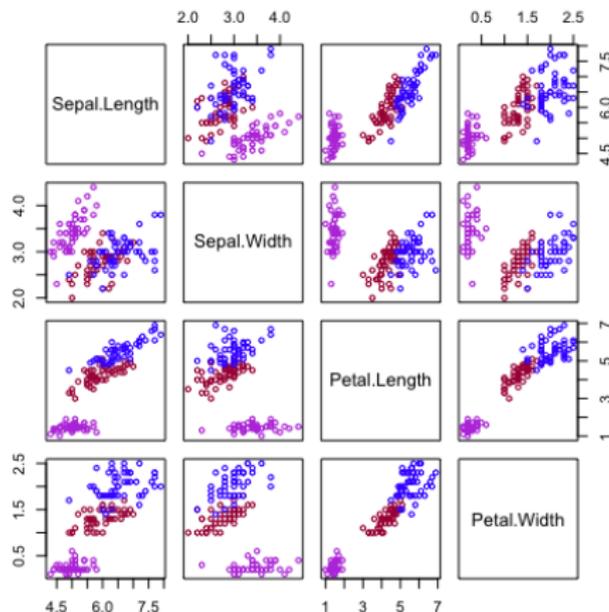
- ▶ par un plot

```
plot(mes.iris)
```



Visualisation des données - coloration par espèces

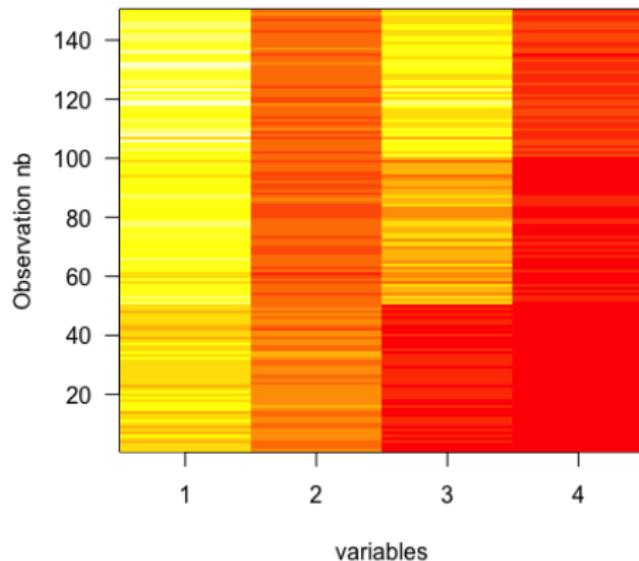
```
species.colors <- c(setosa = "#BB44DD", virginica = "#AA0044",  
                    versicolour = "#0000BB")  
plot(mes.iris, col = species.colors[iris$Species], cex = 0.5)
```



Visualisation des données

- ▶ par la fonction `image()`

```
image(1:nb.var, 1:nb.iris ,t(as.matrix(mes.iris)), xlab = '')
```



Nettoyage des données (1): données manquantes

Avant de commencer à travailler, il est nécessaire de commencer par vérifier que :

- ▶ il n'y a pas de données manquantes

```
sum(is.na(mes.iris))
```

```
[1] 0
```

Nettoyage des données (2) : variables constantes

- ▶ aucune variable n'est constante (aucune variable n'a une variance nulle)

```
iris.var <- apply(mes.iris, 2, var)
kable(iris.var, digits = 3, col.names = "Variance")
```

	Variance
Sepal.Length	0.686
Sepal.Width	0.190
Petal.Length	3.116
Petal.Width	0.581

```
sum(apply(mes.iris, 2, var) == 0)
```

```
[1] 0
```

Normalisation

Afin de pouvoir considérer que toutes les variables sont à la même échelle, il est parfois nécessaire de normaliser les données.

- ▶ soit

- ▶ en centrant (ramener la moyenne de chaque variable à 0)

```
mes.iris.centre <- scale(mes.iris, center=TRUE, scale=FALSE)
```

- ▶ soit

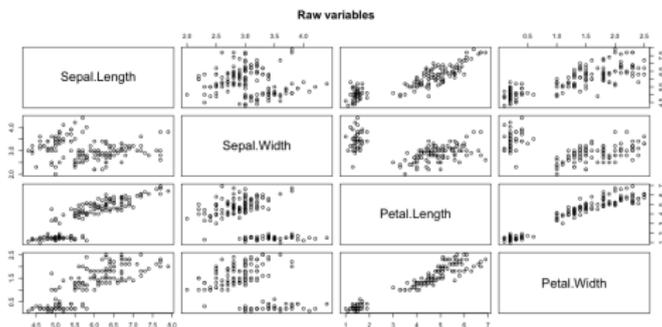
- ▶ en centrant (ramener la moyenne de chaque variable 0)
 - ▶ et mettant à l'échelle (ramener la variance de chaque variable à 1)

```
mes.iris.scaled <- scale(mes.iris, center=TRUE, scale=TRUE)
```

On peut visuellement regarder l'effet de la normalisation :

par un plot des données

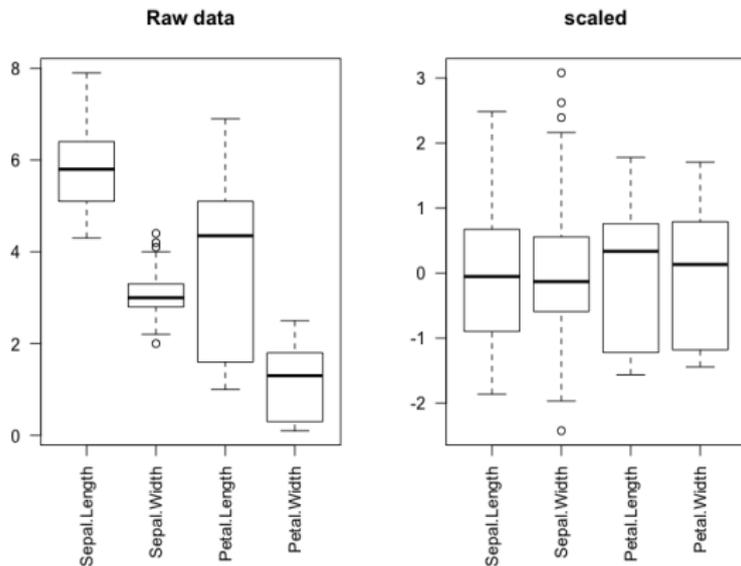
```
plot(mes.iris, main = "Raw variables")
```



! ne pas faire si “grosses” données

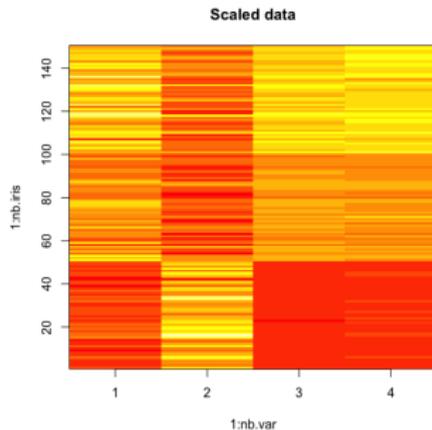
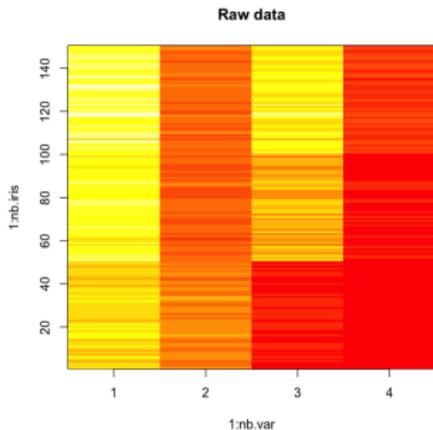
... par une boîte à moustaches (boxplot)

```
par(mfrow = c(1,2))  
par(mar = c(7, 4.1, 4.1, 1.1)) # adapt margin sizes for the  
boxplot(mes.iris, main = "Raw data", las = 2)  
boxplot(mes.iris.scaled, main = "scaled", las = 2)
```



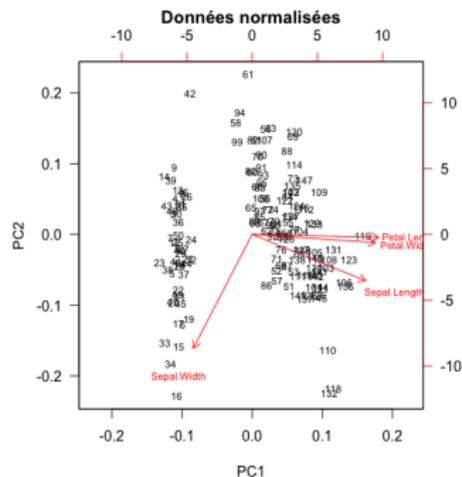
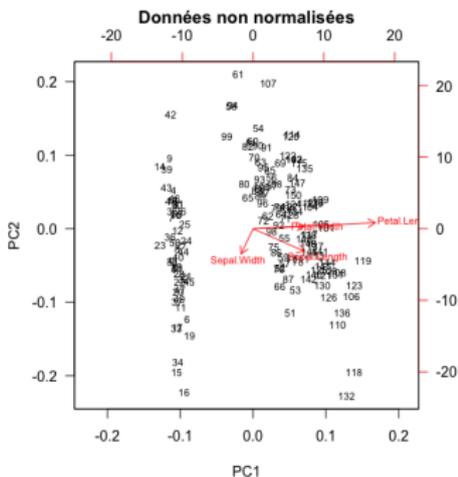
... par une image

```
par(mfrow=c(1,2))  
image(1:nb.var, 1:nb.iris, t(as.matrix(mes.iris)), main="Raw data")  
image(1:nb.var, 1:nb.iris, t(as.matrix(mes.iris.scaled)), main="Scaled data")
```



... par une projection sur une ACP

```
par(mfrow = c(1,2))  
biplot(prcomp(mes.iris), las = 1, cex = 0.7,  
       main = "Données non normalisées")  
biplot(prcomp(mes.iris, scale = TRUE), las = 1, cex = 0.7,  
       main = "Données normalisées")
```

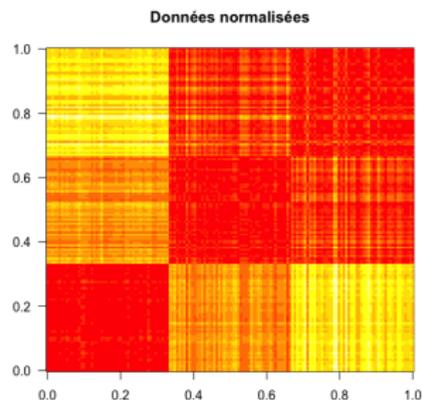
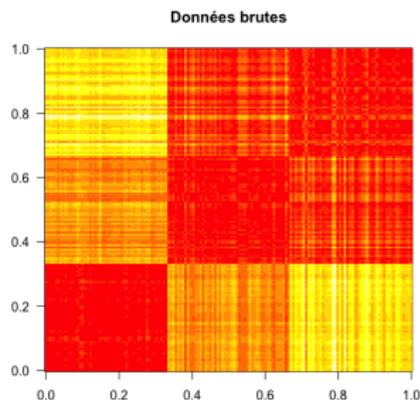


La matrice de distances

Nous utilisons ici la distance euclidienne.

```
iris.euc <- dist(mes.iris)
iris.scale.euc <- dist(mes.iris.scaled)
```

```
par(mfrow = c(1,2))
image(t(as.matrix(iris.euc)), main = "Données brutes", las = 1)
image(t(as.matrix(iris.scale.euc)), main = "Données normalisées", las = 1)
```



La classification hiérarchique

Principe

- ▶ **classification hiérarchique** : mettre en évidence des liens hiérarchiques entre les individus
 - ▶ classification hiérarchique **ascendante** : partir des individus pour arriver à des classes / cluster
 - ▶ classification hiérarchique **descendante** : partir d'un groupe qu'on subdivise en sous-groupes / clusters jusqu'à arriver à des individus.

Notion importante, cf distances

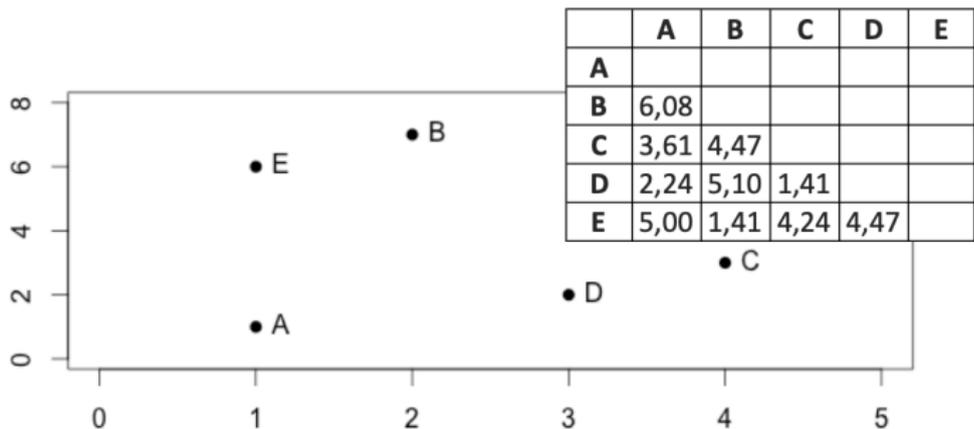
- ▶ ressemblance entre individus = distance
- ▶ ressemblance entre groupes d'individus = critère d'aggrégation
 - ▶ lien simple
 - ▶ lien complet
 - ▶ lien moyen
 - ▶ critère de Ward

L'algorithme

étape 1 :

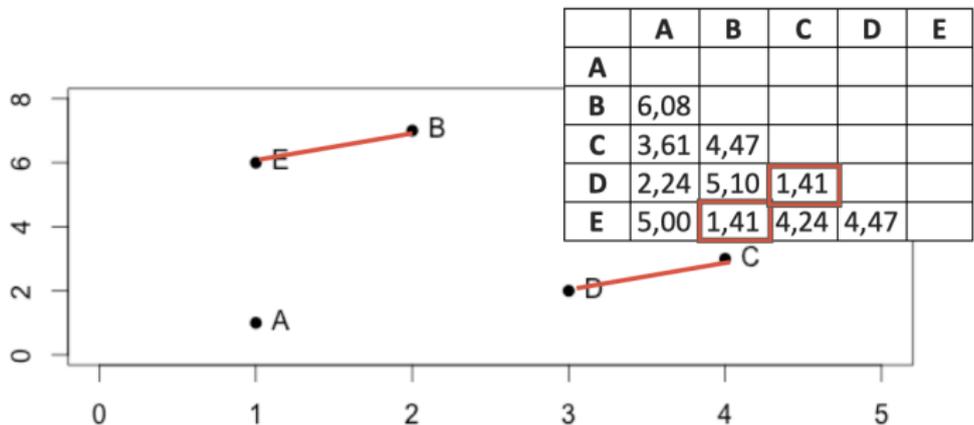
- ▶ départ : n individus = n clusters distincts
- ▶ calcul des distances entre tous les individus
 - ▶ choix de la métrique à utiliser en fonction du type de données
- ▶ regroupement des 2 individus les plus proches $\Rightarrow (n-1)$ clusters

au départ



initialisation : (A), (B), (C), (D), (E)

identification des individus les plus proches



première partition : (A), (BE), (CD)

construction du dendrogramme

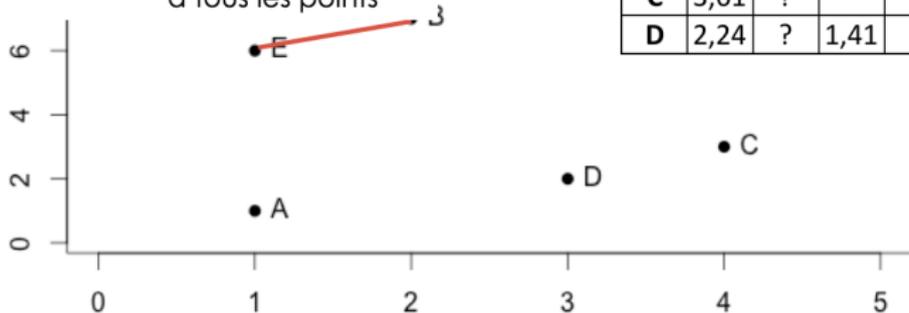


étape j :

- ▶ calcul des dissemblances entre chaque groupe obtenu à l'étape $(j - 1)$
- ▶ regroupement des deux groupes les plus proches $\Rightarrow (n - j)$ clusters

calcul des nouveaux représentants 'BE' et 'CD'

par le lien moyen,
calcul de BE
calcul de la distance de BE
à tous les points



	A	BE	C	D
A				
BE	?			
C	3,61	?		
D	2,24	?	1,41	

par le lien moyen,
calcul de CD
calcul de la distance de CD

calcul des distances de l'individu restant 'A' aux points moyens

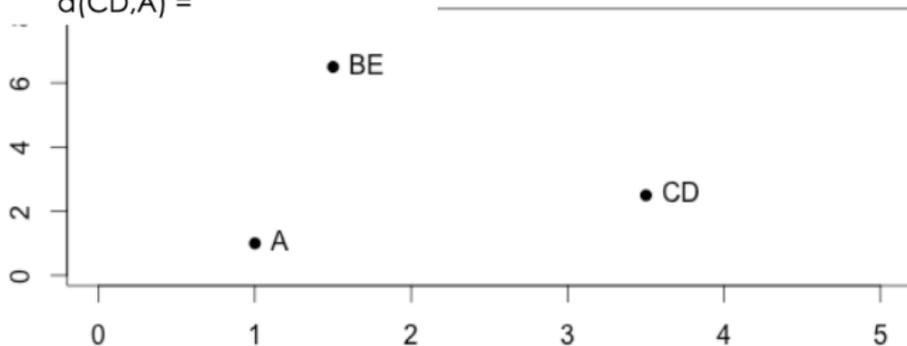
calcul des distances

$$d(\text{BE}, \text{A}) =$$

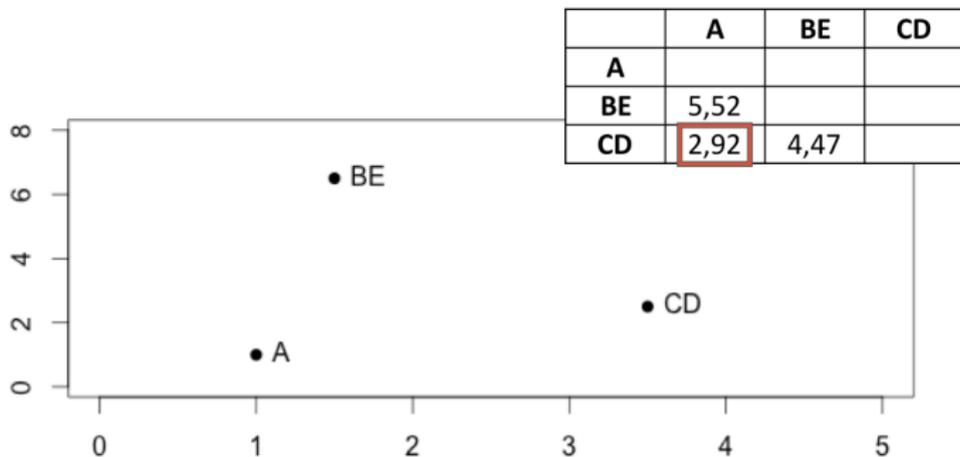
$$d(\text{BE}, \text{CD}) =$$

$$d(\text{CD}, \text{A}) =$$

$$d(\text{BE}, \text{A}) = \frac{d(\text{B}, \text{A}) + d(\text{E}, \text{A})}{2} = \frac{6.08 + 5}{2} = 5.52$$

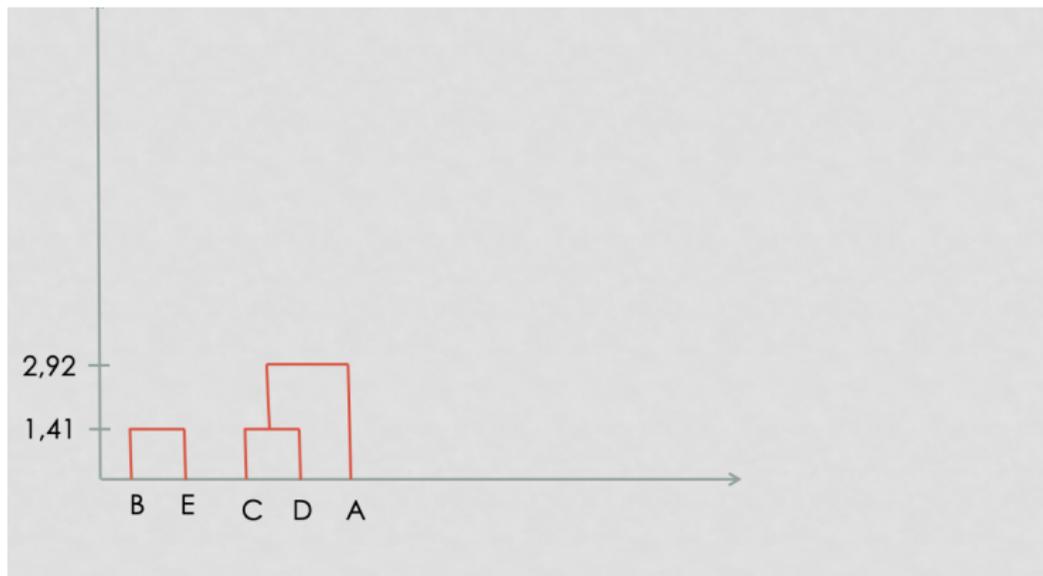


A est plus proche de ...



deuxième partition : (BE), (CDA)

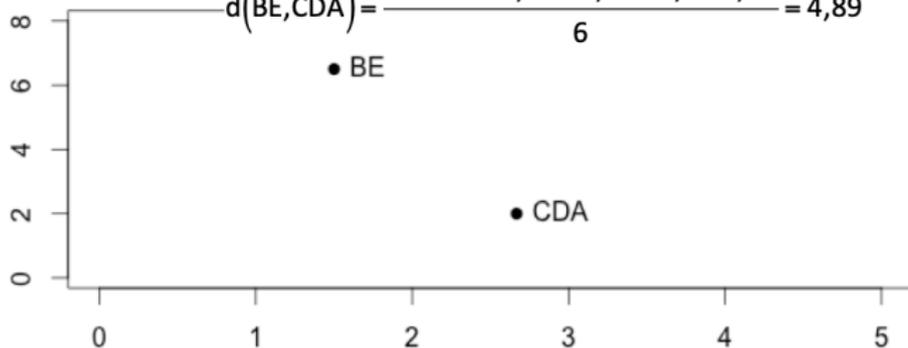
dendrogramme



pour finir

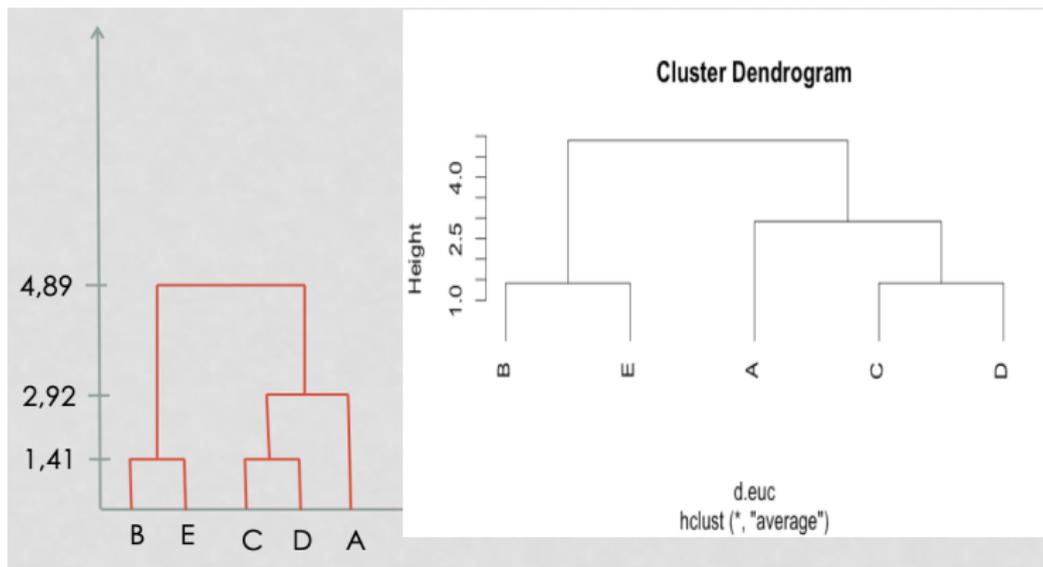
$$d(\text{BE}, \text{CDA}) = \frac{d(\text{B}, \text{C}) + d(\text{E}, \text{C}) + d(\text{B}, \text{D}) + d(\text{E}, \text{D}) + d(\text{B}, \text{A}) + d(\text{E}, \text{A})}{6}$$

$$d(\text{BE}, \text{CDA}) = \frac{6.08 + 5 + 4.47 + 4.24 + 5.1 + 4.47}{6} = 4.89$$



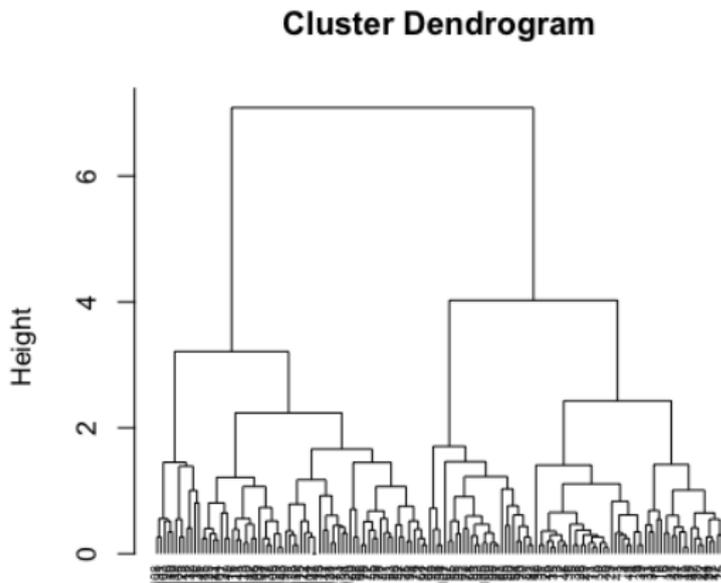
- ▶ à l'étape $(n - 1)$, tous les individus sont regroupés dans un même cluster

dendrogramme final



Je ne fais pas attention à ce que je fais ...

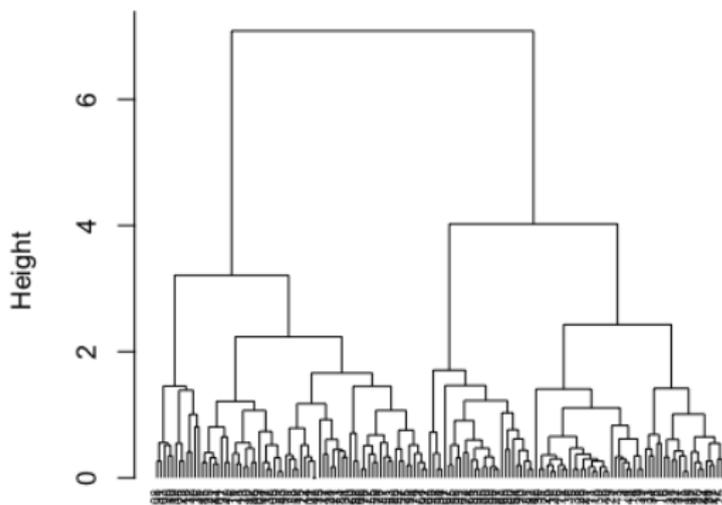
```
iris.hclust <- hclust(iris.euc)  
plot(iris.hclust, hang = -1, cex = 0.5)
```



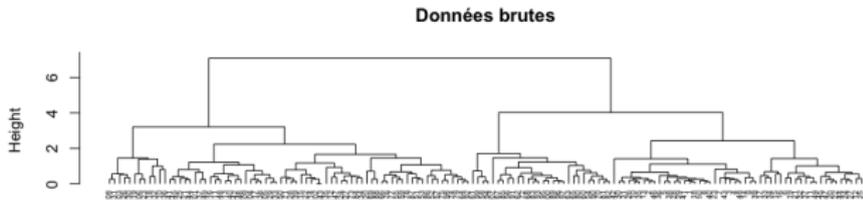
Sur données normalisées

```
iris.scale.hclust <- hclust(iris.scale.euc)  
plot(iris.scale.hclust, hang = -1, cex = 0.5)
```

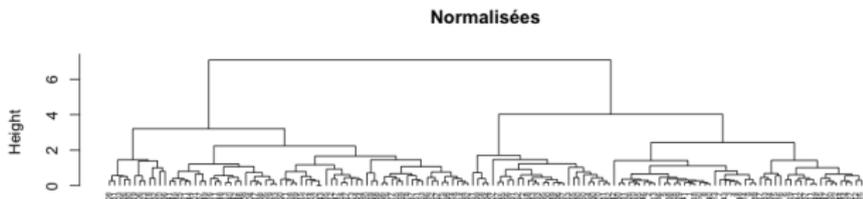
Cluster Dendrogram



```
par(mfrow = c(2, 1))  
plot(iris.hclust, hang = -1, cex = 0.5, main = "Données brutes")  
plot(iris.scale.hclust, hang = -1, cex = 0.5, main = "Normalisées")
```



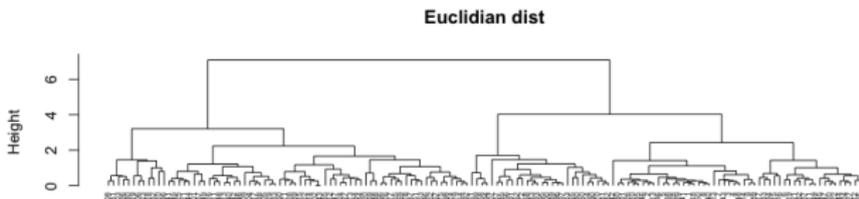
iris.euc
hclust ("complete")



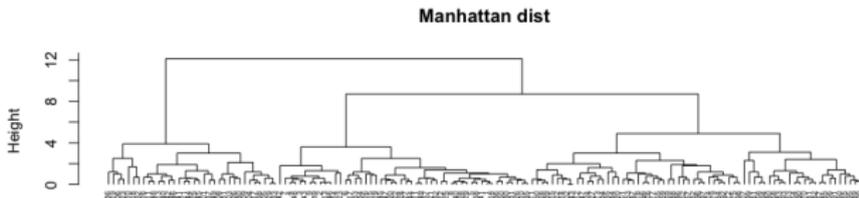
iris.scale.euc
hclust ("complete")

En utilisant une autre métrique

```
iris.scale.max <- dist(mes.iris.scaled, method = "manhattan")
iris.scale.hclust.max <- hclust(iris.scale.max)
par(mfrow=c(2,1))
plot(iris.scale.hclust, hang=-1, cex=0.5, main = "Euclidian")
plot(iris.scale.hclust.max, hang=-1, cex=0.5, main = "Manha
```

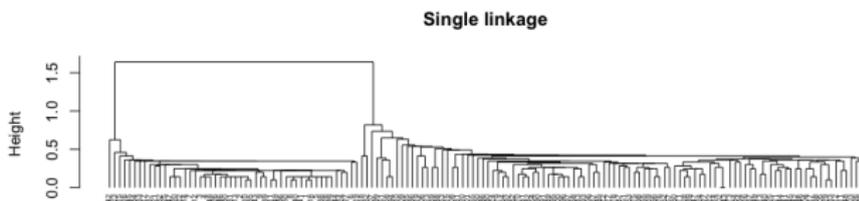


iris.scale.euc
hclust("complete")

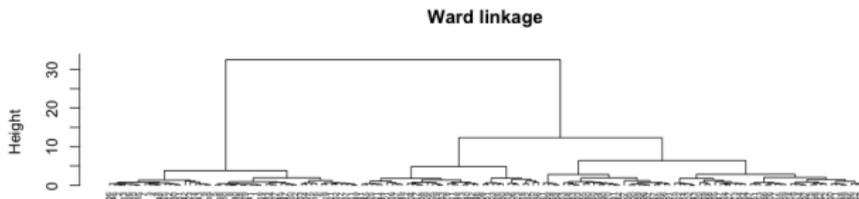


En utilisant un autre critère d'agrégation

```
iris.scale.hclust.single <- hclust(iris.scale.euc, method="single")
iris.scale.hclust.ward <- hclust(iris.scale.euc, method="ward")
par(mfrow=c(2,1))
plot(iris.scale.hclust.single, hang=-1, cex=0.5, main = "Single linkage")
plot(iris.scale.hclust.ward, hang=-1, cex=0.5, main = "Ward linkage")
```

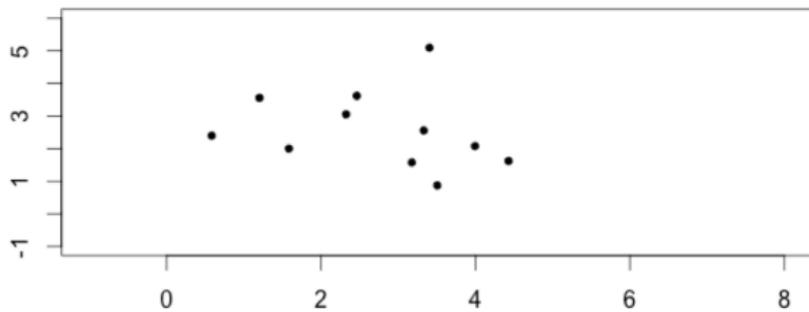


iris.scale.euc
hclust ("single")



Les k-means

Les individus dans le plan



L'algorithme

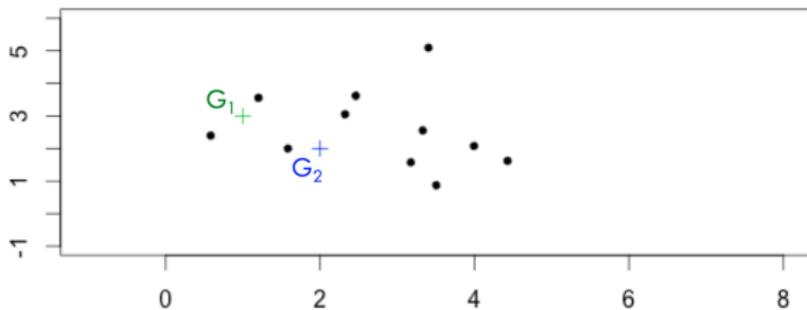
étape 1 :

- ▶ k centres provisoires tirés au hasard
- ▶ k clusters créés à partir des centres en regroupant les individus les plus proches de chaque centre
- ▶ obtention de la partition P_0

Choix des centres provisoires

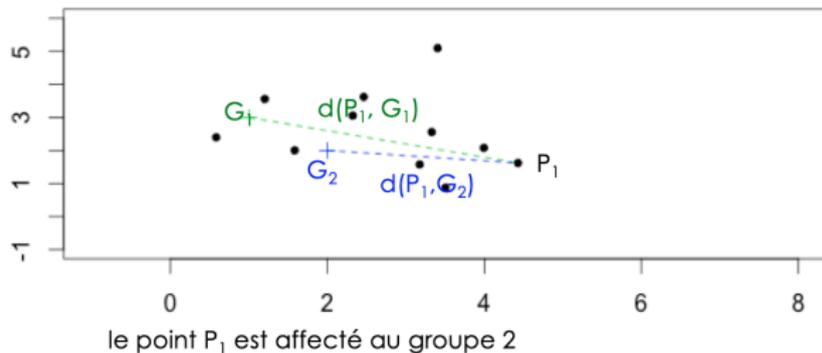
combien de cluster ?

les deux centres initiaux (G_1 et G_2) sont choisis au hasard

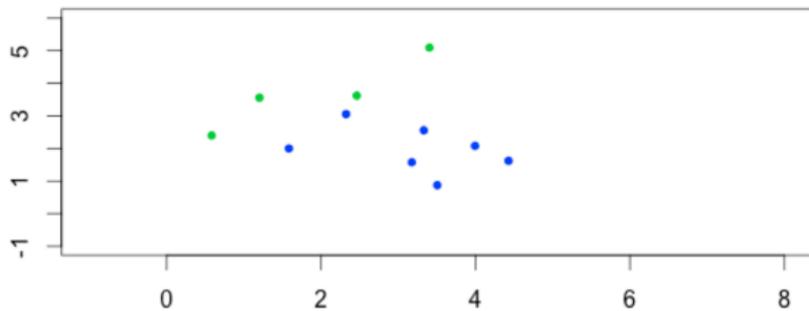


Calcul des distances aux centres provisoires

- calcul des distances de chaque point aux centres G_1 et G_2 ,



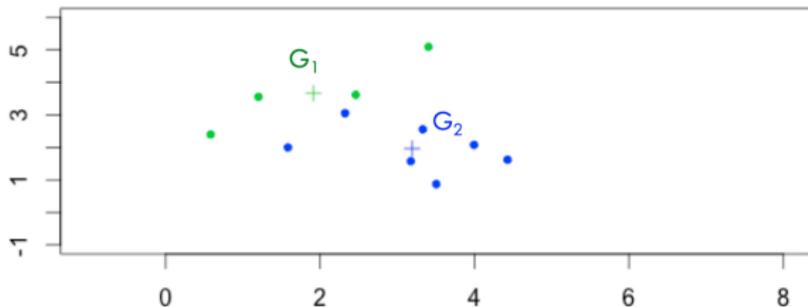
Affectation à un cluster



Calcul des nouveaux centres de classes

Etape j :

- ▶ construction des centres de gravité des k clusters construits à l'étape $(j - 1)$
- ▶ k nouveaux clusters créés à partir des nouveaux centres suivant la même règle qu'à l'étape 0
- ▶ obtention de la partition P_j

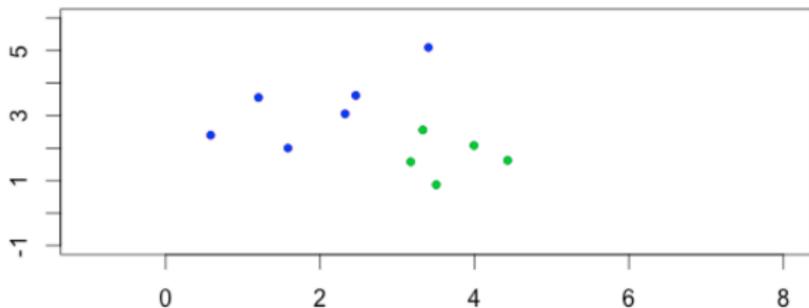


Fin :

- ▶ l'algorithme converge vers une partition stable

Arrêt :

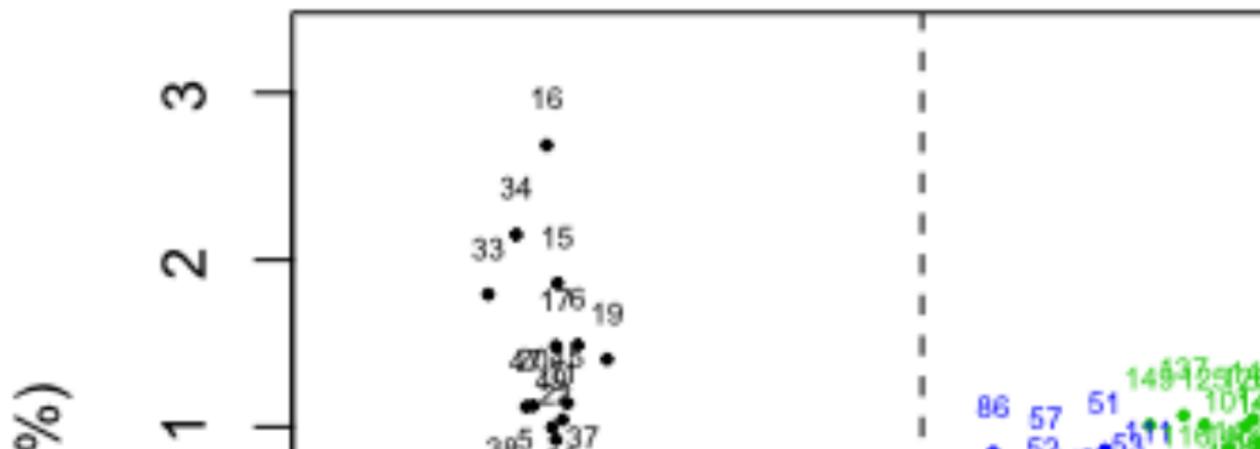
- ▶ lorsque la partition reste la même, ou lorsque la variance intra-cluster ne décroît plus, ou lorsque le nombre maximal d'itérations est atteint.



Visualisation des clusters

```
plot(iris.scaled.acp, col.ind = iris.scale.kmeans5$cluster,
```

Individuals factor map



Combien de clusters ?

Quand une partition est-elle bonne ?

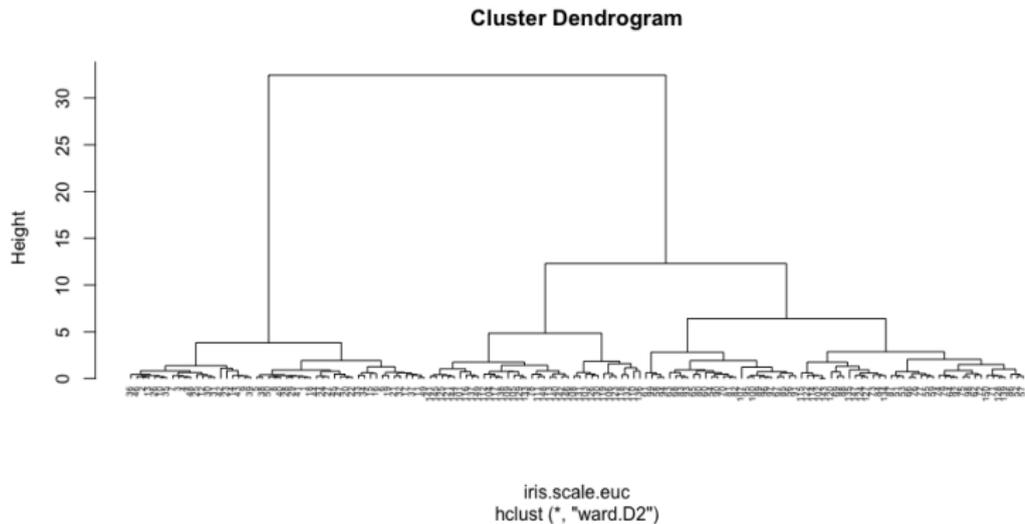
- ▶ si les individus d'un même cluster sont proches
 - ▶ homogénéité maximale à l'intérieur de chaque cluster
- ▶ si les individus de 2 clusters différents sont éloignés
 - ▶ hétérogénéité maximale entre chaque cluster

Classification hiérarchique

La coupure de l'arbre à un niveau donné construit une partition. la coupure doit se faire :

- ▶ après les agrégations correspondant à des valeurs peu élevées de l'indice
- ▶ avant les agrégations correspondant à des niveaux élevés de l'indice, qui dissocient les groupes bien distincts dans la population.

```
plot(iris.scale.hclust.ward, hang=-1, cex=0.5)
```



K-means

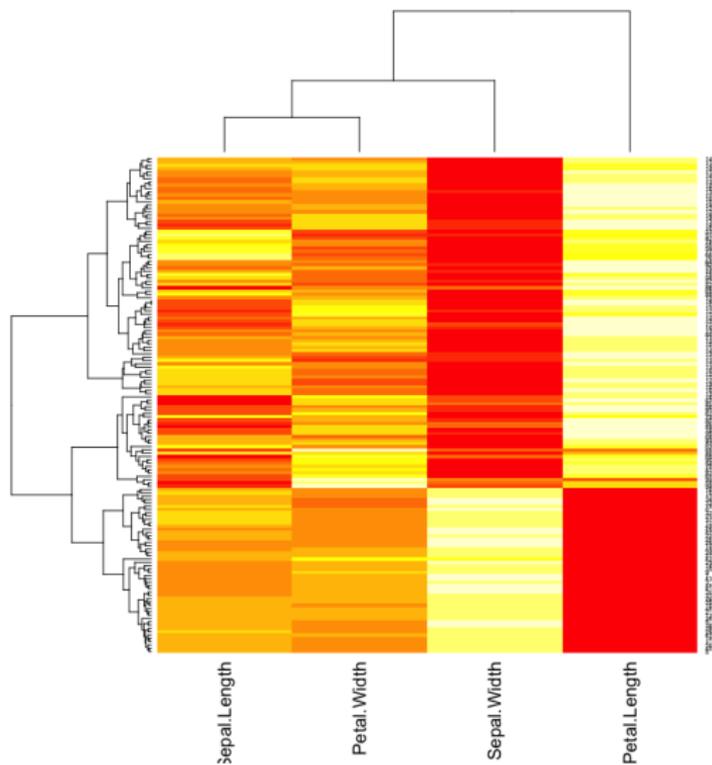
```
I.intra = numeric(length=10)
I.intra[1] = kmeans(mes.iris.scaled, centers=2)$totss
for (i in 2:10) {
  kmi <- kmeans(mes.iris.scaled, centers=i)
  I.intra[i] <- kmi$tot.withinss
}
```

```
plot(1:10, I.intra, type="l")
```

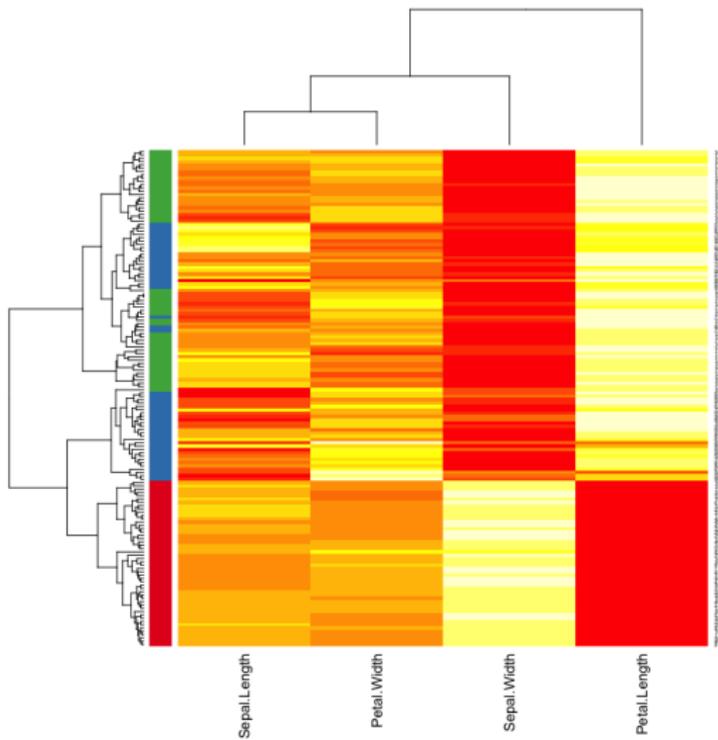


Heatmap

```
heatmap(mes.iris.scaled, margins = c(7,4), cexCol = 1.4, ce
```



```
my_group <- as.numeric(as.factor(substr(variete, 1 , 2)))  
my_col <- brewer.pal(3, "Set1")[my_group]  
heatmap(mes.iris.scaled, RowSideColors = my_col,  
        margins = c(7,4), cexCol = 1.4, cexRow = 0.5)
```



Comparaison de clustering: Rand Index

Mesure de similarité entre deux clustering

à partir du nombre de fois que les classifications sont d'accord

$$R = \frac{m + s}{t}$$

- ▶ m = nombre de paires dans la même classe dans les deux classifications
- ▶ s = nombre de paires séparées dans les deux classifications
- ▶ t = nombre de paires totales

Comparaison de clustering: Adjusted Rand Index

$$ARI = \frac{RI - ExpectedRI}{MaxRI - ExpectedRI}$$

- ▶ ARI=RI normalisé
- ▶ Prend en compte la taille des classes
- ▶ ARI=1 pour classification identique
- ▶ ARI \simeq 0 pour classification aléatoire (peut être <0)
- ▶ Adapté pour nombre de classe différent entre les deux classifications et taille de classe différente

Comparaison des résultats des deux classifications

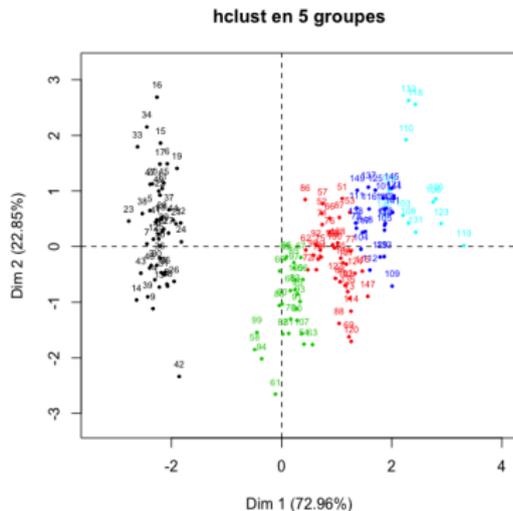
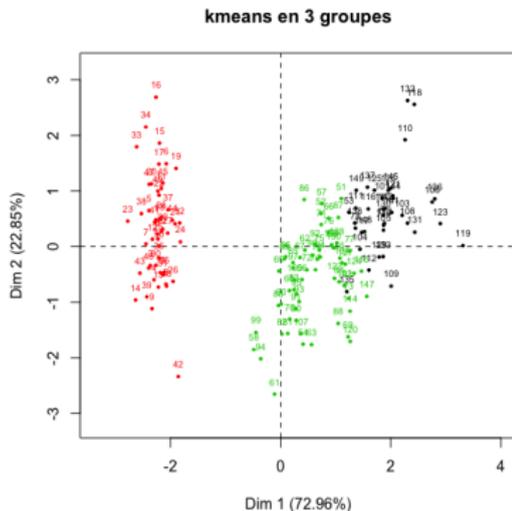
- ▶ par une table de confusion

```
cluster.kmeans3 <- iris.scale.kmeans3$cluster  
cluster.hclust5 <- cutree(iris.scale.hclust.ward, k=5)  
table(cluster.hclust5, cluster.kmeans3)
```

	cluster.kmeans3			
cluster.hclust5	1	2	3	
1	0	0	50	
2	36	2	0	
3	26	0	0	
4	0	24	0	
5	0	12	0	

► par une visualisation

```
par(mfrow=c(1,2))  
plot(iris.scaled.acp, col.ind=cluster.kmeans3, choix="ind")  
plot(iris.scaled.acp, col.ind=cluster.hclust5, choix="ind")
```



```
par(mfrow=c(1,1))
```

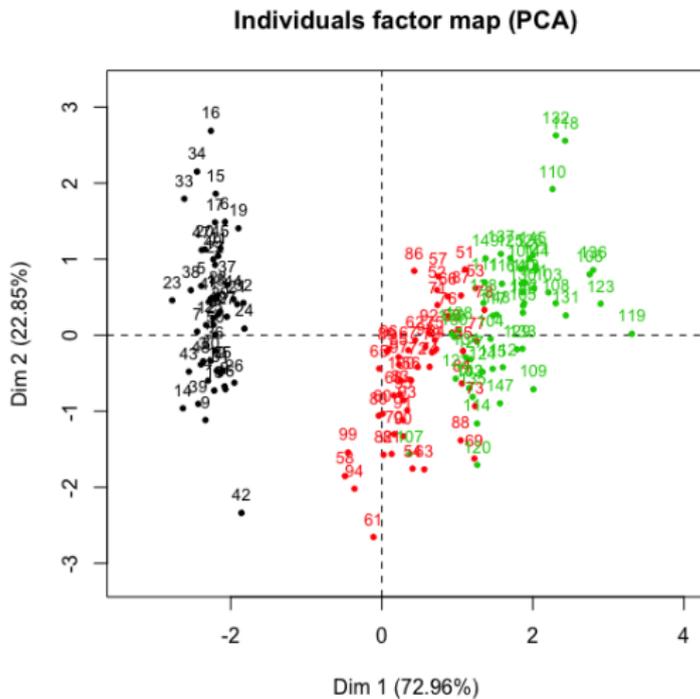
Comparaison avec la réalité

La réalité

```
variete <- iris[,5]  
table(variete)
```

```
variete  
  setosa versicolor  virginica  
      50         50         50
```

```
plot(iris.scaled.acp, col.ind=variete, choix="ind", cex=0.8
```



Comparer k-means avec la réalité

```
conf.kmeans <- table(variete, cluster.kmeans3)  
kable(conf.kmeans, caption = "Confusion table: 3-clusters k-means versus actual class")
```

Table 1: Confusion table: 3-clusters k-means versus actual class

	1	2	3
setosa	0	0	50
versicolor	48	2	0
virginica	14	36	0

Setosa vs others

Visualisation

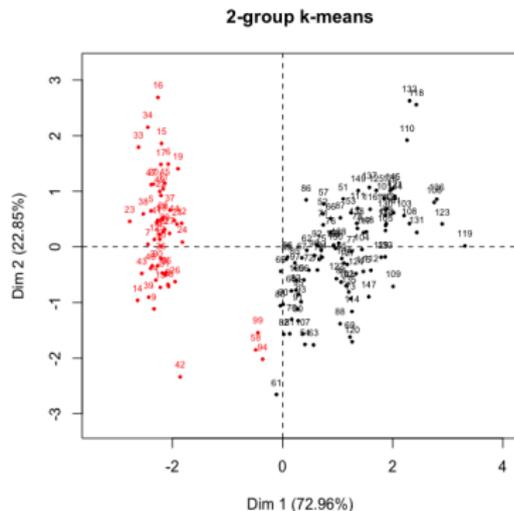
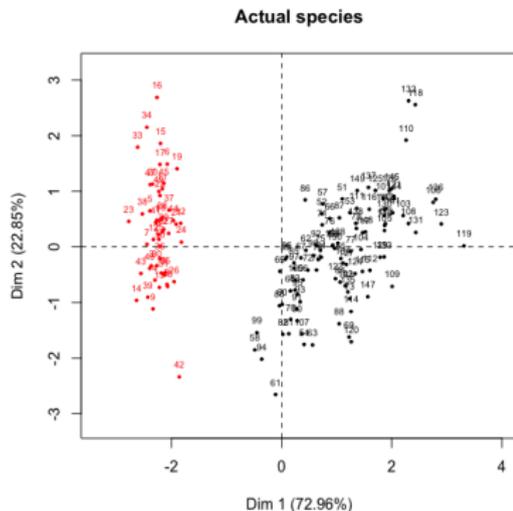
```
variete2 <- rep("notSetosa", 150)
variete2[variete=="setosa"] <- "setosa"
variete2 = factor(variete2)
table(variete2)
```

```
variete2
notSetosa    setosa
          100         50
```

```

par(mfrow=c(1,2))
plot(iris.scaled.acp, col.ind=variete2, title="Actual species
cluster.kmeans2 <- kmeans(mes.iris.scaled, center=2)$cluster
plot(iris.scaled.acp, col.ind=cluster.kmeans2, title="2-gro

```



```

par(mfrow=c(1,1))

```

Table de confusion et calcul de performances

```
conf.kmeans <- table(variete2, cluster.kmeans2)
kable(conf.kmeans)
```

	1	2
notSetosa	3	97
setosa	50	0

- ▶ table de confusion, taux de bien prédits, spécificité, sensibilité,

...

```
TP <- conf.kmeans[1,1]
FP <- conf.kmeans[1,2]
FN <- conf.kmeans[2,1]
TN <- conf.kmeans[2,2]
P <- TP + FN           # nb positif dans la réalité
N <- TN + FP           # nb négatif dans la réalité
FPrate <- FP / N       # = false alarm rate
Spe <- TN / N          # = spécificité
Sens <- recall <- TPrate <- TP / P      # = hit rate ou re
PPV <- precision <- TP / (TP + FP)
accuracy <- (TP + TN) / (P + N)
F.measure <- 2 / (1/precision + 1/recall)
performance <- c(FPrate, TPrate, precision, recall, accuracy)
names(performance) <- c("FPrate", "TPrate", "precision", "recall", "accuracy")
```

```
kable(performance, digits=3)
```

	x
FPrate	1.000
TPrate	0.057
precision	0.030
recall	0.057
accuracy	0.020
F.measure	0.039
Spe	0.000
PPV	0.030

- ▶ rand index et adjusted rand index

```
clues::adjustedRand(as.numeric(variete2), cluster.kmeans2)
```

Rand	HA	MA	FM	Jaccard
0.9605369	0.9204051	0.9208432	0.9639434	0.9302767

Versicolor vs !Versicolor

Visualisation

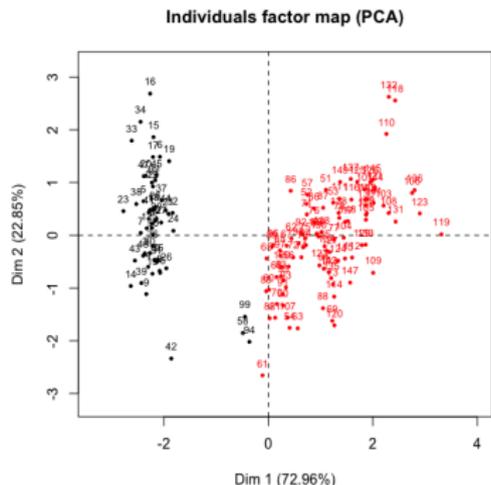
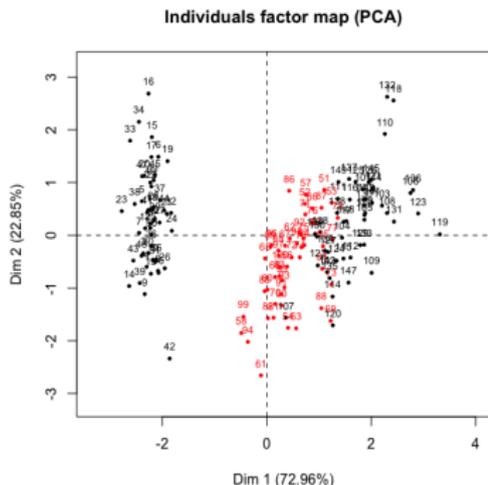
```
variete2 <- rep("notVersicolor", 150)
variete2[variete=="versicolor"] <- "versicolor"
variete2 = factor(variete2)
table(variete2)
```

```
variete2
notVersicolor    versicolor
              100              50
```

```

par(mfrow=c(1,2))
plot(iris.scaled.acp, col.ind = variete2, cex =0.7)
cluster.kmeans2 <- kmeans(mes.iris.scaled, center=2)$cluster
plot(iris.scaled.acp, col.ind = cluster.kmeans2, cex = 0.7)

```



```

par(mfrow=c(1,1))

```

Table de confusion et calcul de performances

```
conf.kmeans <- table(variete2, cluster.kmeans2)
kable(conf.kmeans)
```

	1	2
notVersicolor	50	50
versicolor	3	47

```
TP <- conf.kmeans[1,1]
FP <- conf.kmeans[1,2]
FN <- conf.kmeans[2,1]
TN <- conf.kmeans[2,2]
P <- TP + FN           # nb positif dans la réalité
N <- TN + FP           # nb négatif dans la réalité
FPrate <- FP / N        # = false alarm rate
Spe <- TN / N           # = spécificité
Sens <- recall <- TPrate <- TP / P           # = hit rate ou re
PPV <- precision <- TP / (TP + FP)
accuracy <- (TP + TN) / (P + N)
```

```
kable(performance, digits=3)
```

	x
FPrate	0.515
TPrate	0.943
precision	0.500
recall	0.943
accuracy	0.647
F.measure	0.654
Spe	0.485
PPV	0.500

```
clues::adjustedRand(as.numeric(variete2), cluster.kmeans2)
```

```
          Rand          HA          MA          FM          Jaccard  
0.53995526 0.07211421 0.07722223 0.57895580 0.40737752
```

Pros et cons des différents algorithmes

Algorithme	Pros
Hiérarchique	L'arbre reflète la nature imbriquée de tous les sous-clusters Permet une visualisation couplée dendrogramme (groupes) Choix a posteriori du nombre de clusters (élagage)
K-means	Rapide (linéaire en temps), peut traiter des jeux de données

Contact: anne.badel@univ-paris-diderot.fr